www.cadanda.com Computer-Aided Design AND APPLICATIONS

Surface Pattern Generation Using L-system

Pui Lam Chan[1] and Kin Chuen Hui[2]

[1]Chinese University of Hong Kong, plchan@mae.cuhk.edu.hk
[2]Chinese University of Hong Kong, kchui@mae.cuhk.edu.hk

ABSTRACT

The aim of this research is to create aesthetic patterns automatically on a mesh surface by using 2D L-system. Patterns are generated by providing cross-sections and loci as elements to form tiles. They are then packed together to form local pattern using 2D L-system. Genetic Algorithm is adopted to optimize the patterns. Finally cross-section is blended on the mesh along a locus defined in each tile to form surface. A non-self-intersecting offset surface is generated and its distance from the mesh surface is used as a maximum height to prevent the pattern intersecting itself.

Keywords: aesthetic, pattern, L-system, genetic algorithm, tiling, geometric synthesis.
DOI: 10.3722/cadaps.2013.339-353

1 INTRODUCTION

Geometric synthesis has been a popular topic in computer graphics. Most of the work synthesize geometric textures seamlessly on object surfaces to create ornaments or decorative patterns [2][3][15][16]. However, the input textures are often static and defined by users. It is desirable to see how aesthetic pattern can be generated automatically over the input mesh.

This research generates aesthetic patterns on surface mesh automatically by using L-system and Genetic Algorithm. An automatic 3D aesthetic pattern generator is built. The output of L-system is in the form of 2D array which is the basic building block of the pattern. Each element in the 2D array represents a tile to be put on mesh. Each tile is a basic element of the pattern which contains fundamental information that controls how the pattern should look like. This fundamental information includes a cross-section, a locus and its orientation. The aesthetics of the pattern is optimized by a fitness function in Genetic Algorithm which is based on the measure of symmetry [5], connectivity and complexity of the pattern. Finally, the cross-section is blended [2] along the locus defined in each tile.

Poly-cube map [14] is used to parameterize mesh into seamless square domains that can be directly matched with the square tiles generated by L-system. Besides, we generate non-intersecting

offset surface [17] as a constraint to prevent self-intersection of the pattern. The process is automatic and user is only required to provide cross-sections, loci, their orientations and a generation stage of L-system. The generator will try to fit the pattern correctly onto the input mesh without self-intersection and at while keeping the aesthetic of the pattern.

In Section 2, we briefly go through previous work related to our approach. In Section 3, we show our approach in generating pattern locally using L-system. In Section 4, we describe the implementation of Genetic Algorithm in our approach and a fitness function to optimize the aesthetics value. In Section 5, we review the Poly-cube mapping, surface blending method and show our modifications. In Section 6, we describe the method to control the global pattern and prevent self-intersection. In Section 7, we discuss the strengths and limitations of the approach and give a conclusion in Section 8.

2 BACKGROUND AND RELATED WORK

2.1 L-system and Genetic Algorithm

L-system is a kind of parallel rewriting system introduced by Astrid Lindenmayer [7]. Genetic Algorithm is an evolutionary design algorithm based on a phenomenon of the survival of the fittest organisms in nature. It is used in evolving a given situation and producing a new design based on a given fitness function.

Grammatical Evolution was first proposed by Ryan and O'Neil [8] by combining the shape grammar together with Genetic Algorithm. Gregory S. Hornby and Jordan B. Pollack [9] used L-system as an encoding of Genetic Algorithm to create tables automatically. All axioms and production rules are randomly initialized. Their Genetic Algorithm focuses on evolving the production rules by recombination and mutation. Matthew Lewis [18] introduced a continuous layered pattern function to generate a layered pattern on 2D images by using Genetic Algorithm and tried to extend it into 3D space. His features are based on geometries obtained by interpolating two implicit functions.

2.2 Pattern

2D L-system had been used to generate Virtual Landscape [4] by displacing vertices on a plane. Hokky Situngki [10] used a modified L-system with 2D Thue-Morse sequence to generate traditional Indonesian pattern. Due to the property of Thue-Morse sequence, this method can only generate a fixed 2D matrix. Mark Dow [19] further extended this modified 2D L-system for generating different fractals 2D art images. Nevertheless, the result does not consider the aesthetics of the pattern. Therefore, a careful planning is required to generate aesthetically good images.

In this paper, the fitness function adopted is an extension of the work by Allen Klinger [5] who proposed a numerical method to evaluate the aesthetic interest of a 2D array. His work is based on the Gestalt Principle [6] to measure the symmetry and similarity of elements inside a 2D array. This research extends his method to measure the symmetry, continuity and complexity of the pattern generated on a 3D mesh.

2.3 Texture Synthesis

Yuanyuan Li [3] introduced an approach for generating patterns on a 3D mesh surface. He used a vector field to guide a path produced by L-system. Some components are then placed along the

generated path as ornament. Their work mainly depends on the design and position of vector field. In addition, there is no consideration on the aesthetic value of the pattern.

Craig S. Kaplan [13] worked on mapping 2D semi-regular texture on mesh surface by using poly-cube mapping. His work maps a single 2D texture on the whole mesh seamlessly. He analyzes the importance and requirements of a texture in order to have a seamless cover over the whole mesh.

Kun Zhou [15] introduced mesh quilting which is a 3D texture synthesis algorithm. A shell map is used for storing the texture. Deformation and shape matching are used to synthesize all the texture on the mesh surface.

2.4 Surface Blending

Gershon Elber [2] proposed using a given 2D cross-section $Cs(r) = (x_S(r), y_S(r))$ to blend two surfaces. $(x_S(r), y_S(r))$ is the 2D coordinate of a given cross-section Cs. His idea is based on the use of Hermite curve to achieve G^1 continuity. Given a curve $C(t) = (u(t), v(t))$ on a parametric surface $S(u,v)$, two rail curves $C_d(t)$ and $C_{-d}(t)$ are constructed. These two curves are approximate offset curves of $C(t)$ with d being the offset distances. Let $n(u(t), v(t))$ be the normal field of the surface, then two tangent fields are there along $C_d(t)$ and $C_{-d}(t)$:

$$T_d(t) = C'_d(t) \times n(u(t), v(t)),\ T_{-d}(t) = C'_{-d}(t) \times n(u(t), v(t))$$

An ornament pattern can be generated across the rail curves $C_d(t)$ and $C_{-d}(t)$ by:

$$S(r,t) = A_0(t) + D_0(t)x_S(r) + n(u(t), v(t))y_S(r) + T_{-d}(t)h_{10}(r) + T_d(t)h_{11}(r) \quad (1)$$

$$\text{where } D_0(t) = \frac{(C_d(t) - C_{-d}(t))}{2},\ A_0(t) = \frac{(C_d(t) + C_{-d}(t))}{2}$$

$(x_S(r), y_S(r))$ is the x-y coordinate of the cross-section. To ensure G1 continuity, the end points of the cross-section should be duplicated so that they are equal to $(x_S(0), y_S(0)) = (-1,0)$ and $(x_S(1), y_S(1)) = (1,0)$. On the other hand, $h_{10}(r)$ and $h_{11}(r)$ are the blending functions of the Hermite curve $r(r-1)^2$ and $r^2(r-1)$ respectively.

3 PATTERN GENERATION

3.1 L-system based Pattern Generation

3.1.1 Tile

Each tile contains a cross-section, a locus, and its orientation as elements. Using different combination of elements can form different patterns.2D L-system is used to generate the combination of elements

in a tile by subdividing a grid [4] into cells (such as 2x2, 3x3, 4x4, etc.). The grid is in a square domain between (0, 0) to (1, 1). Besides, a cross-section index, a locus index and its orientation are stored in each tile.

3.1.2 Axiom and Rules

To simplify the representation, we use a string to define our 2D L-system. Each character represents a specified locus and cross-section. For example,

L-system: {ABAA.ABBA.ABAA}

Rules: A $\rightarrow$ ABAA, B $\rightarrow$ ABBA

Axiom: ABAA

This string represents information about a 2D L-system and it is separated by dots. The first two parts represent two rules and the last one is the axiom that is an initial status of the pattern. Each part can be of different dimensions such as 1x1, 2x2, 3x3 etc. The total number of available alleles is computed by the following expression.

$$\text{Number of alleles} = \text{number of path} \times \text{number of cross-section} \times \text{number of orientation} \quad (2)$$

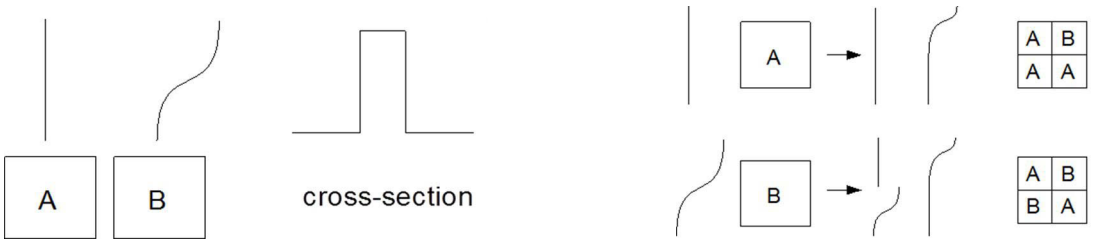

Fig. 1: An illustration of a 2D L-system: (a) Loci and cross-sections (b) Rules.

For example, assume there are two paths named as A and B and one cross-section named as C_1. Meanwhile, for each locus, we represent its four orientations in the form of $(A_0, A_{90}, A_{180}, A_{270})$. Therefore, this L-system will have eight alleles. Graphical representation of the results of L-system {ABAA.ABBA.A} is shown in Fig.2. We can see that the pattern has a fractal property. Fig.3 shows another L-system with different orientations used as alleles.

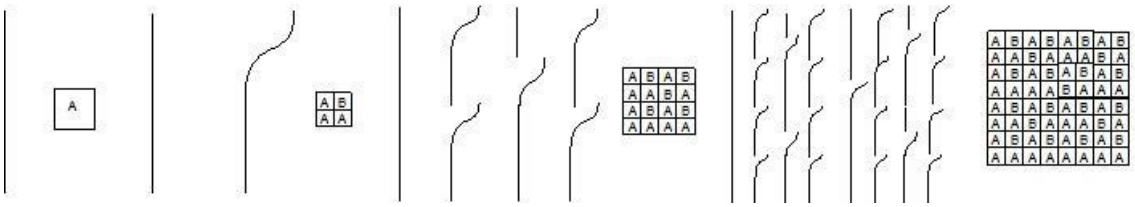

Fig. 2: An example of a 2D L-system: (a) 1st generation (b) 2nd generation (c) 3rd generation (d) 4th generation.

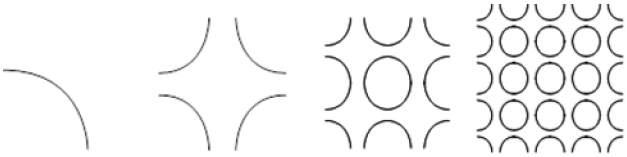

Fig. 3: 2D L-system includes alleles with different orientation: (a) 1st generation (b) 2nd generation (c) 3rd generation (d) 4th generation.

4 AESTHETIC VALUE COMPUTATION

4.1 Genetic Algorithm

The figures below illustrate the workflow of a pattern generation process using our 2D L-system with Genetic Algorithm. First, we initialize a population of L-system with the number of rules equal to the number of alleles. In this way the L-system can be expanded without termination and every allele can be involved in the growing process. User then defines a generation of the L-system to control the details of the pattern and the L-system will grow up to this generation. In every generation, a fitness function is used to evaluate the aesthetics. A predefined threshold is used to select the best L-system and each L-system will be evaluated to get a score. If the score is not higher than a threshold, genetic operators are performed on the best parents in the population to produce new offspring and the whole process is repeated. If the best score is higher than the threshold, the process will be terminated and the best L-system will be the final pattern.

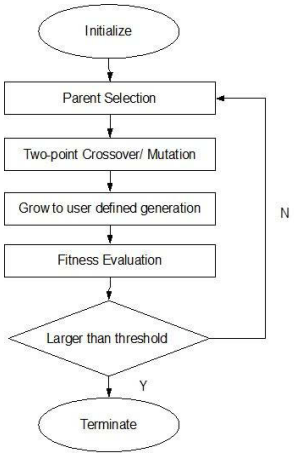

Fig. 4: Workflow of 2D L-system combined with Genetic Algorithm.

4.1.1 *Initialization*

The L-system strings are randomly initialized. Rules and axioms are randomly generated from the alleles.

4.1.2 *Operators*

Two operators are used and they are two-point crossover and mutation. Two-point crossover exchanges part of the string between two L-systems. Mutation randomly replaces an element inside a string with other elements in the alleles. They are performed on both rules and axioms.

Two point crossover: {AAAB.ABAB.BABB} → {ABBB.ABAB. BABB}

{ABBB.ABAB.ABAA} → {AAAB.ABAB.ABAA}

Mutation: {ABAB.ABBB.BAAA} → {ABAB.ABBB.ABBB}

4.1.3 *Pattern measurement*

4.1.3.1 *Symmetry*

The measurement of symmetry is similar to the pattern measure method described in [5]. However the whole 2D array is considered and only the first six symmetries are needed as the measurement for harmonic, they are:

I. 90, -90 rotation symmetry
II. 180 rotation symmetry
III. x axis symmetry
IV. y axis symmetry
V. (x, y) axis symmetry
VI. (-x, y) axis symmetry

f_H is defined for symmetry measure where H counts the items above.

$$f_H = \frac{H}{6} \tag{3}$$

Considering only the strings of 2D L-systems below is difficult to decide whether it is symmetric about the (-x,-y) axis. It can be done by comparing end points of the loci that contain the same letters.

Fig. 5: Two patterns: (a) $BA_{90}A_{90}B$ (asymmetric) and (b) $BA_{180}A_{90}B$ (symmetric).

4.1.3.2 *Complexity*

A similar approach to the complexity measurement of image is described in [20]. A 2D array is divided into R rows and C columns. If two consecutive cells inside that 2D array are different, we increase K_c or K_R by one. Finally, the complexity is calculated by:

$$f_{Com} = \frac{K_R + K_c}{R \times (C-1) + C \times (R-1)} \tag{4}$$

4.1.3.3 *Connectivity*

We define K to be the number of endpoints inside each cell. An endpoint is "connected" if there is another endpoint with the same 2D coordinates exists inside any neighboring cells (Fig. 6). We define C to be the number of connected endpoints and N to be the number of cells. The connectivity is measured by Eq. (5) and the value is between (0, 1).

$$f_{Con} = \frac{C}{N \times K} \tag{5}$$

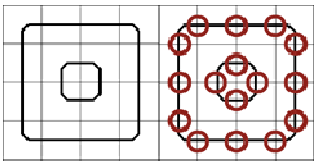

Fig. 6: (a) A pattern and (b) the connection points in each cell.

4.1.3.4 *Fitness function*

The fitness function used in this paper is defined as

$$fitness = \frac{(a \times f_H + b \times f_{Con} + c \times f_{Com})}{(a+b+c)} \tag{6}$$

where a, b and c are weights. Since the symmetry Eq. (3), connectivity Eq. (5) and complexity Eq. (4) are always between [0, 1], the fitness function also lies between [0, 1].

Fig. 7: Examples of local pattern generated on parametric surfaces.

5 SURFACE BLENDING

5.1 Parameterization

5.1.1 Overview of inverse Poly-Cube Map

The algorithm of Poly-Cube Map is described in [14]. It provides a seamless texture mapping method for square texture. Based on this algorithm, we try to map our 3D patterns seamlessly on a mesh surface. The original algorithm first maps the 3D coordinate to 3D texture coordinate. The 3D texture coordinate is projected onto the face-lets of the poly-cubes, and finally they are mapped to a 2D domain to determine the pixel on the texture using a Look-Up Table.

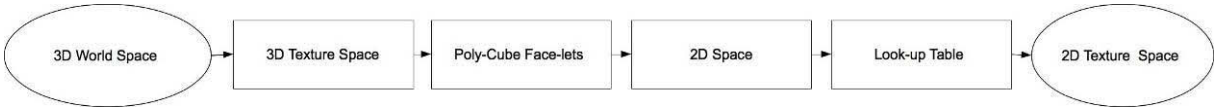

Fig. 8: Steps of Poly-cube mapping.

In our implementation, we do everything backwards. First we generate the pattern and use the loci as the texture coordinates. Face-lets of the cubes are all mapped to 2D domain between (0, 0) and (1, 1). We map the texture coordinate to the face-lets position. Using this face-lets position we project it back to the 3D texture coordinate by using Ray-Triangle intersection. Finally, we use barycentric coordinates of the triangles to determine points in the 3D space.

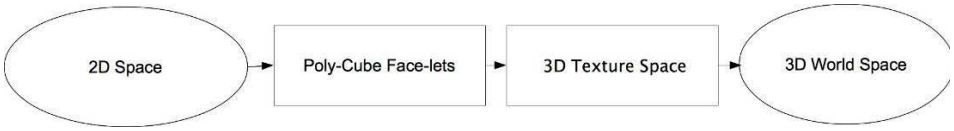

Fig. 9: Steps of our mapping.

5.1.2 Face-lets mapping

We use the loci in the 2D array generated by L-system as texture coordinate. The face-lets are first mapped onto 2D square domain from (0, 0) to (1, 1). The loci coordinates are then mapped on to every Poly-Cube face-lets.

5.1.3 3D texture mapping

We follow the mapping as described in [14] to obtain the 3D texture coordinates of each mesh vertex. We maintain faces and edges in 3D texture space according to the mesh topology in 3D space.

5.1.4 Inverse projection to 3D world space

According to the algorithm of Poly-Cube Map described in [14], a dual space of unit cubes is constructed in 3D texture space. Mesh in 3D texture space is divided into many unit cubes. After that each vertex of the cube becomes the center of another group of identical virtual unit cubes. Those virtual cubes refer to the Poly-cube and are classified into six different types where each type has its own projection direction. In [14], any vertex in 3D texture space is projected along its direction defined by the types of the virtual cube. In this research, we inversely map any point on the face-let of the virtual cube back to the 3D texture coordinate by using the directions in Table 1 according to different

types. Ray- Triangle intersection is used to find the intersection point p on the 3D texture space. We separate Type 5 into three extra types (Type 5a, 5b and 5c) in order to have a correct projection. Given a point p=(r, s, t) on the face-lets of a cube and the coordinates are between (0, 0, 0) and (1, 1, 1), the projection vector are shown in Table 1.

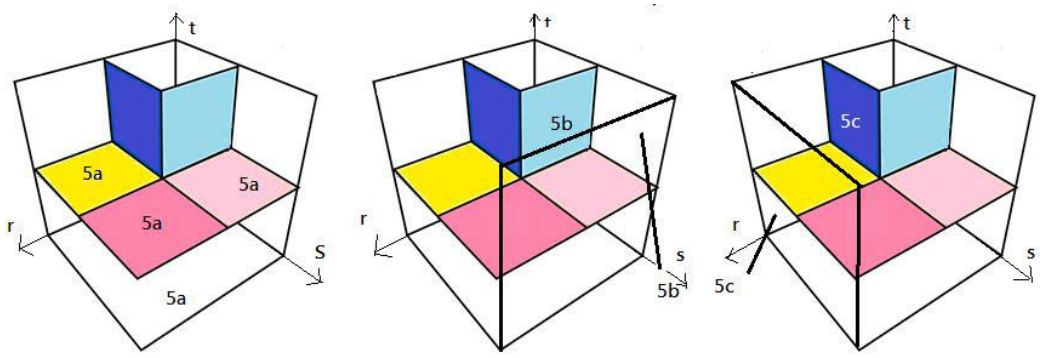

Fig. 10: (a) Type 5a Poly-Cube Face-let (white) and its corresponding faces (red, yellow and pink). (b) Type 5b (c) Type 5c.

Type	3	4a	4b	5a
Projection Direction	(-r,-s,-t)	(-r,0,-t)	(0,0,-1)	(r-$\frac{r}{s}$,s-1,t-1) if $s \geq r$ (r-1,s-$\frac{s}{r}$,t-1) if s<r

Type	5b	5c
Projection Direction	(-r,-0.5,t-1)	(-0.5,-s,t-1)

Tab. 1: The projection directions of Type 3 to Type 5c are shown. They are the opposite directions mentioned in [14].

After projection we parameterize each face in 3D texture space into its 2D domain. We obtain the local 2D coordinates of the face vertices, and hence the 3D texture coordinate of the point p, the actual 3D coordinate, normal and tangents by using the barycentric coordinate of each triangle.

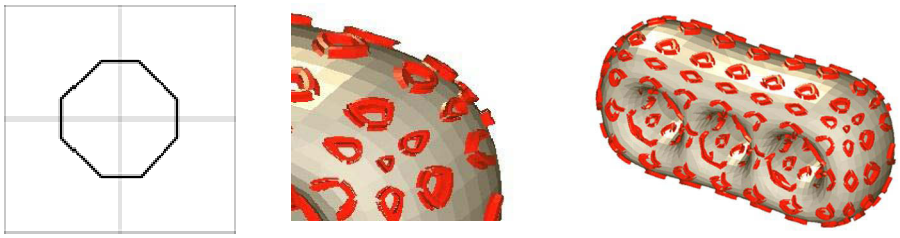

Fig. 11: (a) and (b) Basic local pattern generated by our L-system is shown. (c) Repeated patterns generated on mesh surfaces using Poly-cube mapping with our L-system. (d)(e)(f) Another example.

5.2 Extended Surface Blending

Gershon Elber's [2] work considers two boundary curves. Here we modify his work by using Coons Patch such that the generated ornamental can be fit onto the four boundary curves of the surface inside our tile in the pattern. Let the starting point of the path be (x_0, y_0) and end point be (x_1, y_1). We parameterize (x, y) into $t(x, y)$ such that $t(x_0, y_0) = 0$ and $t(x_1, y_1) = 1$. We define our path in a tile as $C(t(x_p, y_p))$, where (x_p, y_p) are the coordinates of the locus. Two rail curves $C_d(t(x_p, y_p))$ and $C_{-d}(t(x_p, y_p))$ are constructed. To simplify the representation, we denote $C(t(x_p, y_p))$ as $C(t)$. By [2] and Eq. (1), we have the blended surface $S(r,t) = S_1 + S_2$.

$$S_1 = A_0(t) + D_0(t)x_S(r) + n(u(t), v(t))y_S(r) \tag{7}$$

$$S_2 = T_{-d}(t)h_{10}(r) + T_d(t)h_{11}(r) \tag{8}$$

where $D_0(t) = \dfrac{(C_d(t) - C_{-d}(t))}{2}$ and $A_0(t) = \dfrac{(C_d(t) + C_{-d}(t))}{2}$.

We expand S_1 Eq. (7) and rephrase it to get:

$$S_1 = q(r)C_d(t) + (1 - q(r))C_{-d}(t) + n(u(t), v(t))y_S(r) \qquad \text{where } q(r) = \frac{(1 + x_S(r))}{2} \tag{9}$$

$q(r)$ is a value between (0, 1).We define two curves $M(q(r),0)$ and $M(q(r),1)$ which connect the starting points and end points of both $C_d(t)$ and $C_{-d}(t)$ (i.e. $C_d(0)$ and $C_{-d}(0)$ or $C_d(1)$ and $C_{-d}(1)$) in (x_p, y_p) space. For lines connect $C_d(t)$ and $C_{-d}(t)$, we denote them as $M(q(r),t)$. Given a value $t_0(x_0, y_0)$, in order to get the world coordinate of $M(q(r),t_0)$, we can connect $C_d(t_0)$ and $C_{-d}(t_0)$ in world space and get the distance of $C_d(t_0)C_{-d}(t)$. Finally, we get the world coordinate by

$$M(q(r),t_0) = (1-q(r))C_{-d}(t_0)+q(r)C_d(t_0).$$

Alternatively, we can get the position of (x_p,y_p) by the offset curve $C_{d\,q(r)}(t(x_p,y_p))$ and project on the 3D texture space to get the 3D coordinate as described in 5.1.1.We define a new parameter $d_{q(r)}$ where $d_{q(r)}$ = - d when $q(r)$ =0 and $d_{q(r)}$ = d when $q(r)$ =1.The relationship of offset distance $d_{q(r)}$ and $q(r)$ is

$$d_{q(r)} = (2d \times q(r)) - d \tag{10}$$

By Coons Surface, our S1 becomes:

$$S_1 = q(r)C_d(t)+(1-q(r))C_{-d}(t)+tM_1(q(r))+(1-t)M_0(q(r))-T(r,t)+n(u(t),v(t))y_S(q(r)) \tag{11}$$

where $T(r,t) = (1-q(r))(1-t)C_{-d}(0)+t(1-q(r))C_{-d}(1)+(1-t)q(r)C_d(0)+tq(r)C_d(1)$

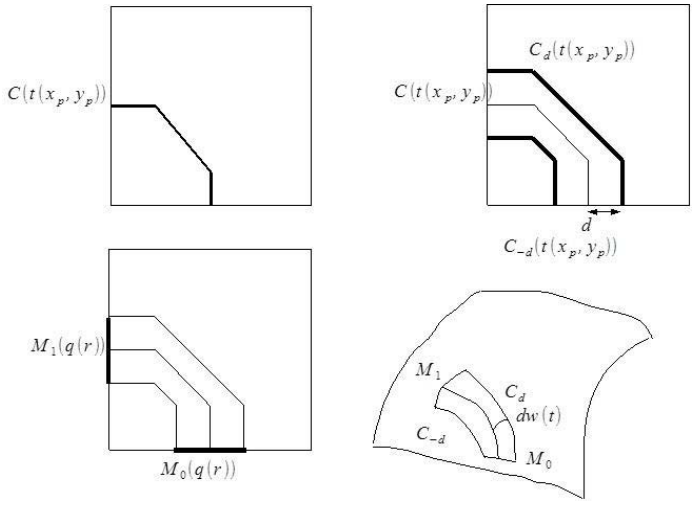

Fig. 12: (a)(b)(c) An illustration of four boundary curves in (x_p,y_p) parametric space and (d) world space.

6 GLOBAL OPTIMIZATION

6.1 Global Pattern

The work in [13] shows that if any texture has a signature of p4 type, the pattern should meet each other at the corner point on the surface. A signature of p4 type is defined as:

- The center of the pattern itself is the center of four-fold rotation.
- The middle points of four edges are the center of two-fold rotation.
- The corners of the pattern are the center of four-fold rotation.

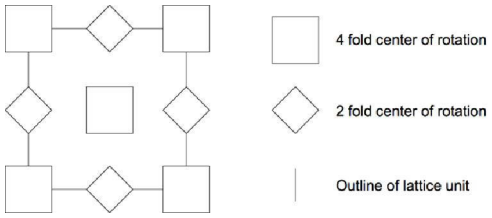

Fig. 13: An illustration of a p4 type texture.

We add these criteria to our fitness function to generate type p4 pattern by L-system. We measure the rotational symmetry at the four corners by dividing a k x k square tile into four $\left(\frac{k}{2}\right)\times\left(\frac{k}{2}\right)$ sub tiles if k is even or four $\left(\frac{k}{2}-1\right)\times\left(\frac{k}{2}-1\right)$ sub-tiles if k is odd. By rotating the sub-tiles around the center of the rotation in clockwise or anti-clockwise direction, we can compare the elements in each sub-tile to check if they are equivalent. Similar comparison can be done to check the two-fold rotation on the edge.

6.2 Dimensions

The dimension of the square tile is denoted as $k\times k$, where k>=1 corresponds to the generation in our 2D L-system. This is similar to the subdivision of basis texture. Finer details are generated on any coarse mesh when the generation of L-system and k increase.

6.3 Self-intersection

3D pattern generated on a mesh may exhibit self-intersection therefore a process has to be performed to prevent. We generate an offset surface based on the method described in [17]. An offset surface is first generated by displacing each vertex by $\frac{1}{10}$ of the diagonal of the bounding box along its normal. We use an Octree structure and detect if there is any intersection between faces inside any cube of the Octree. For those faces that are intersected, we revert their vertices to their original positions and decease the step size for the next run. This process is repeated until there is no face intersection. Fig. 14, Fig.15.

The offset surface can be used as a reference of the pattern's maximum height. According to [2], Eq. (11) can be further extended to

$$S_1 = q(r)C_d(t) + (1-q(r))C_{-d}(t) + tM_1(q(r)) + (1-t)M_0(q(r)) - T(r,t) + n(u(t),v(t))y_S(q(r))V(t) \quad (12)$$

where $V(t)$ is a scaling which refers to the height of the ornament. We set the distance between the offset surface and the mesh surface as $V(t)$. Since all the ornament should be generated along the face normal $n(u(t),v(t))$, they should not intersect each other.

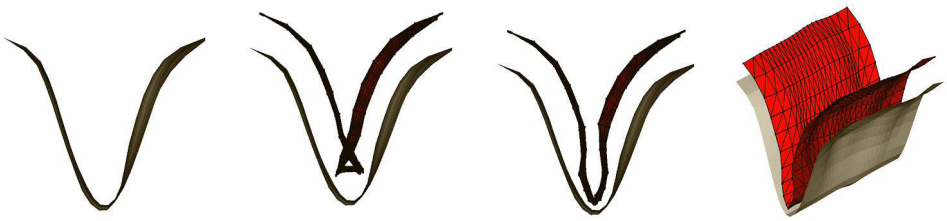

Fig. 14: (a) A mesh surface (b) A self-intersecting offset surface (red) generated. (c) and (d) The intersections of offset surface are avoided after repeating the process described in 6.3.

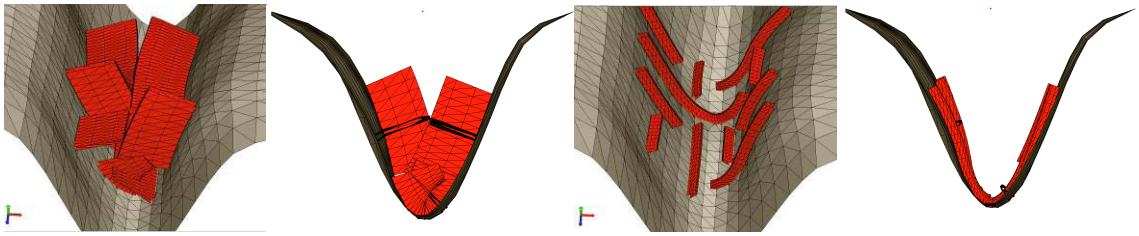

Fig. 15: (a) and (b) show the patterns (red) generated on the input surface before constraining the maximum height of pattern to be the minimum distance between the offset surface and the input surface. (c) and (d) The result after constraining.

6.4 Variation

Instead of applying the same L-system throughout the whole mesh, we can apply different L-system to selected face-lets of the Poly-cube. In addition, L-system can be used to control the color symmetry of the pattern so as to increase the aesthetic value.

7 DISCUSSION AND FUTURE WORK

In the proposed method, users are required to specify the cross-sections and loci as the basic elements. It is possible to replace the input elements with functions described in [18].

Our L-system only considers the aesthetic value of the local pattern inside a tile. Global pattern is ensured by restricting the local pattern to be of type p4 and repeatedly applying the same pattern. We control the maximum height of the pattern throughout the entire mesh to prevent self-intersection of the pattern.

However, in some cases, we may want to have different types of patterns on the mesh. These patterns may be different in size, height and loci comparing with the original pattern. In this case, the global aesthetic value of the mesh may be affected. In addition, a transition change between patterns will also be another concern.

We may use another L-system G to optimize the global pattern. Assume we use N sub L-system L= {L0, L1 é LN} to generate N local patterns P= {P0, P1 é PN}. All the face-lets of the Poly-cube are packed together to form a rectangle grid and we apply L-system G to the grid. Now each allele of the global L-system G refers to the local patterns in P. This may increase the number of global patterns that can be generated but this will also increase the generation time.

Besides, we may classify our mesh into eight regions by using the segmentation method described in [12] including peak, ridge, saddle ridge, flat, minimal surface, pit, valley and saddle valley. Then we can recognize regions that are concave (pit, valley and saddle valley) and place special patterns in order to maintain aesthetics or prevent intersections.

8 CONCLUSION

In this research, an automatic pattern generator has been developed. It requires users to specify cross-sections and loci as input to generate pleasing pattern automatically by using L-system. The pattern can be easily applied to any parameterized surface. We have extended Poly-cube method to generate pattern seamlessly on mesh. We have also extended a surface blending technique based on Coons Patch. The height of the pattern is constrained by constructing an offset surface to prevent self-intersection. This height indicates the maximum non-self-intersecting distance from the surface.

ACKNOWLEDGEMENT

This work is partially supported by a grant from the University Grant Council of the Hong Kong Special Administrative Region (No. 412508) and a Direct Grant (No. 2050492) from The Chinese University of Hong Kong.

REFERENCES

[1] Matthew, L.: Interactively Evolving Virtual Environment Maps with Continuous Layered Pattern Functions, Proceeding CA '02 Proceedings of the Computer Animation.
[2] Gershon, E.: Generalized Filleting and Blending Operations towards Functional and Decorative Applications, Journal Graphical Models - Special issue on SMI 2003 67(3), May 2005.
[3] Li, Y.: Geometry Synthesis on Surfaces Using Field-Guided Shape Grammars, IEEE Transactions on Visualization and Computer Graphics, 17(2), 2011, 231-243. DOI:10.1109/TVCG.2010.36
[4] Ashlock, D.; Gent, S.; Bryden, K.: Evolution of L-system for compact virtual landscape generation, in Evolutionary Computation, 2005. The 2005 IEEE Congress on, 3, 2005, 2760ð2767.
[5] Klinger, A.: A pattern measure, Environment and Planning B: Planning and Design, 27, 2000, 537-547. DOI:10.1068/b2676
[6] Wertheimer, M.: Laws of Organization in Perceptual Forms, In A Source Book of Gestalt Psychology, W.D. Ellis, ed. (London: Routledge and Kegan Paul).
[7] Lindenmayer, P.: The algorithmic beauty of plants, Springer Verlag, New York,1990
[8] O'Neill, M.; Ryan, C.: Grammatical Evolution. Evolutionary Automatic Programming in an Arbitrary Language, Kluwer Academic Publishers, 2003
[9] Gregory, S.; Jordan, B.: The Advantages of Generative Grammatical Encodings for Physical Design, Congress on Evolutionary Computation.2001
[10] Situngkir, H.: The computational generative patterns in Indonesian Batik, Working Paper Series BFI, WP-5-2008, Bandung Fe Institute, 2008
[11] Wannarumon, S.; Bohez, E.L.J.: A New Aesthetic Evolutionary Approach for Jewelry Design, Computer-Aided Design and Applications, 3(1-4), 385-394, 2006.
[12] Wang, J.; Yu, Z.: Surface feature based mesh segmentation. Computers & Graphics 35(3), 2011, 661-667, Shape Modeling International (SMI) Conference 2011
[13] Kaplan, C. S.: Semiregular patterns on surfaces ,Proceedings of the 7th International Symposium on Non-Photorealistic, Animation and Rendering 2009
[14] Tarini, M.: PolyCube-Maps, Proceeding SIGGRAPH '04 ACM SIGGRAPH 2004 Papers

[15] Zhou, K.: Mesh quilting for geometric texture synthesis, Proceeding SIGGRAPH '06 ACM SIGGRAPH 2006 Papers
[16] Porumbescu: Shell maps, ACM Transactions on Graphics 23, 3, 626ð633
[17] Cohen, J.: Simplification Envelopes. Proceeding SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques
[18] Lewis, M. W.; Parent, R. E.: Interactively Evolving Virtual Environment Maps with Continuous Layered Pattern Functions, CA 2002, 49-54
[19] Dow M., http://lcni.uoregon.edu/~dow/#Research .
[20] Wannarumon, S.; Bohez, E.; Annanon, K.: Aesthetic evolutionary algorithm for fractal-based user-centered jewelry design, AI EDAM 22(1), 19-39 (2008)